RepFusion: Leveraging Multimodal Priors for Denoising in Representation Space

Xichen Pan[1,2] **Aashu Singh**[1] **Satya Narayan Shukla**[1*]
Shlok Kumar Mishra[1†] **Saining Xie**[2†]
[1]Meta AI [2]New York University
https://xichenpan.com/repfusion

Abstract

Large language models (LLMs) are widely used in text-to-image (T2I) systems, but they are typically limited to text encoding, while denoising is handled by newly trained generative backbones. The emergence of representation autoencoders (RAEs) shifts the generation target toward semantically structured visual representations, creating a latent space that is more compatible with pretrained LLM priors. Inspired by multimodal LLMs (MLLMs), where an MLP projector is sufficient to align clean visual representations with a pretrained LLM, we repurpose the MLLM itself as a noisy representation encoder, extending this mechanism from clean to noisy inputs. We present RepFusion, which uses the resulting MLLM outputs as the conditioning signal for a diffusion transformer. In controlled comparisons at similar inference budgets, RepFusion outperforms baselines that devote comparable capacity to newly initialized denoisers. Reusing MLLM outputs across denoising steps further accelerates inference while retaining performance comparable to per-step MLLM recomputation. These results demonstrate that MLLMs provide strong priors for denoising visual representations.

1 Introduction

Text-to-image (T2I) generation is commonly formulated as conditional image generation, where image generators are conditioned on the outputs of text encoders. Alongside the evolution of image generators from GANs [14] to diffusion models [18], text encoders have also progressed from LSTMs [19] to CLIP [34] and T5 [35]. Recently, many systems have replaced these encoders with large language models (LLMs) [3, 44, 16] due to their stronger representational capacity, richer world knowledge, in-context learning ability, and compatibility with unified multimodal models [31]. However, in recent pipelines [7, 55, 48, 46, 24, 4], LLMs still primarily act as static text encoders that produce text embeddings, while diffusion transformers (DiTs) [32] carry out the denoising trajectory and image synthesis.

This division of labor made sense in the VAE [22] era. Diffusion models typically denoise VAE latents, and these latents were never designed to be "read" by pretrained language priors. They are low-dimensional, local, and optimized for reconstruction rather than semantics. As a result, even if one aims to bring an LLM closer to the denoising loop, it is unclear what the LLM should consume or why doing so would be beneficial.

Representation autoencoders (RAEs) [53] change this picture. By moving generation from VAE latents to semantically structured visual representations, such as CLIP [34] or DINO [5] features, RAEs provide a denoising space that is both easier to optimize and more semantically meaningful.

*Work done at Meta.
†Equal advising.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

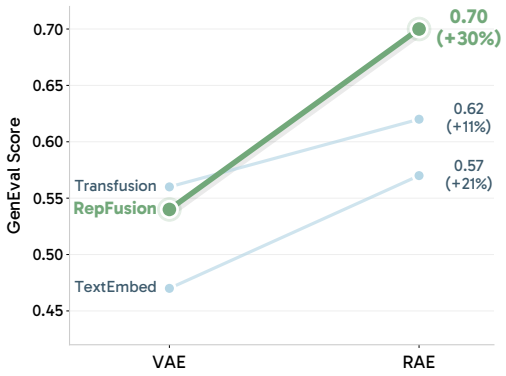


Figure 1: GenEval comparison when switching from VAEs to RAEs for three conditioning strategies: TextEmbed (conditioning a DiT with an LLM's last-layer text token embeddings following recent T2I practice [48, 46, 24, 4]), Transfusion [54], and `RepFusion`. All three variants in this comparison use a 7B LLM, TextEmbed and `RepFusion` also use a 1.3B DiT. `RepFusion` feeds noisy visual representations into a pretrained MLLM and uses the resulting outputs to condition a DiT. It benefits most significantly from the transition, achieving a 30% relative gain (+0.16 absolute improvement), compared to 21% (+0.10) for TextEmbed and 11% (+0.06) for Transfusion.

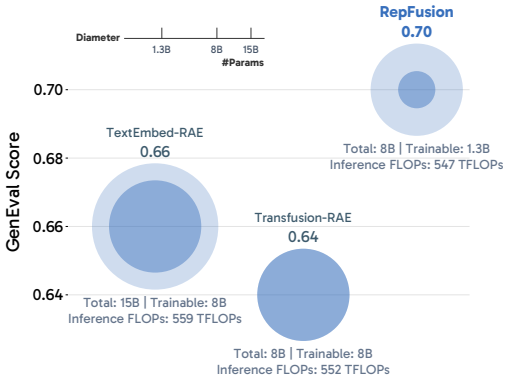


Figure 2: GenEval comparison across different conditioning strategies under similar inference FLOPs. Circle size denotes total parameters, and the inner disk denotes trainable parameters. Each method allocates roughly 8B parameters to modules that either process noisy visual latents or denoise them. `RepFusion` fine-tunes only a 1.3B DiT and an MLP projector, yet outperforms TextEmbed and Transfusion, both of which train 8B parameters (a larger DiT and an LLM, respectively). This cross-method comparison suggests that MLLMs provide strong priors for denoising visual representations, and that repurposing them to encode noisy representations can be a more effective use of parameters than scaling newly initialized denoisers.

Furthermore, these developments bridge T2I and the feature spaces currently utilized by Multimodal LLMs (MLLMs).

In multimodal understanding, pretrained LLMs have demonstrated a simple yet powerful property: with an MLP projector, they can ingest clean visual representations and become strong sequence models over multimodal tokens [26]. This observation is usually discussed in the context of understanding and reasoning. Here, we take it as a design principle for generation: if an LLM can perceive clean visual representations, can it also process noisy counterparts during denoising?

Our answer is yes. As shown in Figure 1, the resulting system is highly effective and best suited to the RAE latent space. We present `RepFusion`, a T2I model that treats a pretrained MLLM as a *noisy representation encoder*. In addition to text inputs, we feed noisy RAE latents into an off-the-shelf MLLM by reusing its MLP projector. We keep the pretrained LLM backbone frozen and fine-tune only its projector. We then use the MLLM's output to condition a DiT that denoises in the same latent space. Conceptually, this design allows the pretrained MLLM to focus on what it does best: modeling structured visual representations.

This design shifts capacity allocation beyond the standard "make the denoiser bigger" recipe. As shown in Figure 2, under similar inference FLOPs, all compared systems allocate roughly 8B parameters to modules that either process noisy visual latents or denoise them: TextEmbed uses a 7B frozen MLLM text encoder and an 8B DiT, Transfusion uses an 8B joint denoising transformer, and `RepFusion` uses the same 7B frozen MLLM together with a 1.3B DiT. We provide details on training the TextEmbed and Transfusion baselines in Appendix B. Despite fine-tuning only the DiT and an MLP projector, `RepFusion` outperforms these baselines, showing that, across model families, allocating substantial model capacity to a frozen pretrained conditional encoder can outperform spending nearly the entire parameter budget on newly initialized denoising modules. This suggests that pretrained MLLMs carry priors that transfer beyond multimodal understanding: once the representation space is compatible, those priors can directly help denoise noisy visual representations.

We also compare against unified architectures such as Transfusion [54], which can be viewed as another way of exposing noisy visual information to language models. As shown in Figure 1, even when we upgrade such baselines to operate in the RAE latent space, the gains are smaller than those obtained by explicitly repurposing a frozen MLLM as a noisy encoder. In other words, moving from VAE to RAE helps, but by itself it does not unlock the full benefit of pretrained language priors.

During training, the MLLM and DiT in `RepFusion` operate at the same timestep. Our default sampling procedure follows this design by running both models at every denoising step. To reduce inference cost, we examine whether the MLLM condition needs to be recomputed at every step. We uniformly divide the denoising steps into segments, running the MLLM once at the beginning of each segment and reusing its outputs while the DiT continues to run at every step. In our recomputation experiments, three MLLM updates over 30 denoising steps retain performance comparable to per-step recomputation, with a $2.4\times$ sampling speedup.

In summary, this paper argues for a simple shift in perspective: many modern T2I systems already allocate substantial capacity to huge LLM text encoders, and RAEs provide a representation space where these encoders can do more than encode text. By letting frozen MLLMs take noisy visual representations as input, we obtain a strong and efficient prior for denoising in representation space. The main contributions are:

- We show that frozen pretrained MLLMs can encode noisy RAE latents and provide useful denoising priors beyond static text conditioning.
- We show that allocating parameters to a frozen pretrained conditional encoder can outperform static text embedding baselines with comparable capacity in newly initialized denoisers.
- We show that the pretrained MLLM prior is strong: freezing it outperforms further jointly optimizing it for generation.
- We show that reusing MLLM outputs across denoising steps accelerates inference while retaining performance comparable to per-step MLLM recomputation.

2 Related work

Text encoders in T2I Early conditional GANs use small text encoders such as LSTMs [19], producing either global sentence embeddings [36, 52] or token-level embeddings [50]. Diffusion models later standardized text conditioning with frozen pretrained encoders that provide token embeddings for cross-attention. Stable Diffusion 1.5 [37] popularized a CLIP [34] text encoder. Recent systems increasingly scale the text encoder: Imagen [38] moves beyond CLIP to LLMs such as T5-XXL [35], and PixArt-α [7], Stable Diffusion 3 [12], and FLUX.1 [23] follow to ship with large T5-family encoders. Recent open-source models such as Lumina-Next [55] and Sana [48] adopt LLM encoders, and FLUX.2 [24] further scales the LLM to a 24B-parameter Mistral Small 3 [42]. Overall, modern T2I pipelines often devote billions of parameters to text encoders, motivating methods that better utilize their capacity.

From VAEs to RAEs Latent diffusion [37] popularized a key design choice in modern T2I models: instead of diffusing in pixel space, models denoise in the latent space of an autoencoder, making high-resolution generation tractable. Most systems adopt VAEs [22] for this purpose, but VAE latents are heavily compressed and optimized for reconstruction, which limits their semantic expressiveness. RAEs [53] avoid this bottleneck by pairing a decoder with a frozen pretrained encoder (e.g., CLIP [34] or DINO [5]), working with semantically rich latents that are easier to denoise. This shift removes the VAE bottleneck and brings T2I into representation spaces that pretrained MLLMs already handle well, creating a natural opportunity to leverage their priors beyond static text conditioning.

Integration of Language Models and Denoisers A growing line of work seeks tighter integration between conditional encoders and denoisers. Unified architectures such as Transfusion [54] train a large transformer to jointly model language outputs and denoise VAE latents, aiming for a single multimodal stack. Another direction builds compact interfaces between MLLMs and diffusion backbones via learnable queries [31, 6, 43] or joint attention [39, 10]. In contrast, our focus is not on the conditioning mechanism, but on changing the content of the condition itself. We push MLLMs beyond text encoding, repurposing them to encode noisy representations and condition DiTs [32].

3 RepFusion

This section first formalizes diffusion in visual representation space (Section 3.1) and describes how RepFusion uses an MLLM to encode noisy representations for DiT conditioning (Section 3.2). We then use controlled ablations to isolate the role of noisy representation inputs (Section 3.3) and multimodal perception pretraining (Section 3.4). Finally, we break down how these ingredients improve over TextEmbed and Transfusion baselines (Section 3.5). Unless otherwise specified, the variants discussed in this section use a 7B LLM backbone paired with a 1.3B DiT denoiser.

3.1 Preliminary

A flow matching T2I model is a conditional generative model. Given a text prompt y, we first obtain a text embedding $\boldsymbol{c} = E_\phi(y)$ with a typically frozen text encoder. The generative network is then conditioned on $\boldsymbol{c}$, either through cross attention [37] or adaptive normalization [32]. In our setting, diffusion operates in a visual representation space: let $\boldsymbol{x}$ denote a clean visual representation, let t be the timestep, and let $\boldsymbol{\epsilon}$ be the Gaussian noise. We adopt the $\boldsymbol{v}$-prediction parameterization [25, 27, 1]: $\boldsymbol{z}_t = t\,\boldsymbol{x} + (1-t)\,\boldsymbol{\epsilon}$, $\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})$. We follow the timestep shifting strategy of Zheng et al. [53], Esser et al. [12]. For a base dimension n and an effective data dimension m, the sampled timestep $t_n \sim \mathcal{U}(0,1)$ is shifted to $t = \frac{\alpha t_n}{1+(\alpha-1)t_n}$, where $\alpha = \sqrt{m/n}$. Following Zheng et al. [53], Esser et al. [12], we use $n = 4{,}096$ and set m to the effective dimensionality of the visual representation; in our RAE setup, this gives $\alpha = 12$.

The flow velocity is given by the time derivative of $\boldsymbol{z}_t$: $\boldsymbol{v} = \boldsymbol{z}_t' = \boldsymbol{x} - \boldsymbol{\epsilon}$. We learn a conditional velocity field $\boldsymbol{v}_\theta(\boldsymbol{z}_t, t, \boldsymbol{c})$ by minimizing the standard flow matching objective [25, 1]:

$$\mathcal{L} := \mathbb{E}_{t,\boldsymbol{x},\boldsymbol{\epsilon}} \left\| \boldsymbol{v}_\theta(\boldsymbol{z}_t, t, \boldsymbol{c}) - \boldsymbol{v} \right\|^2,$$

where $\boldsymbol{v}_\theta$ is predicted by the diffusion model.

3.2 Methods

In standard approaches, the conditioning $\boldsymbol{c}$ relies solely on the text y. In RepFusion, as shown in Figure 3, we augment the conditioning to also include the noisy visual representations, $\boldsymbol{z}_t$. This design allows the LLM to perceive the denoising trajectory.

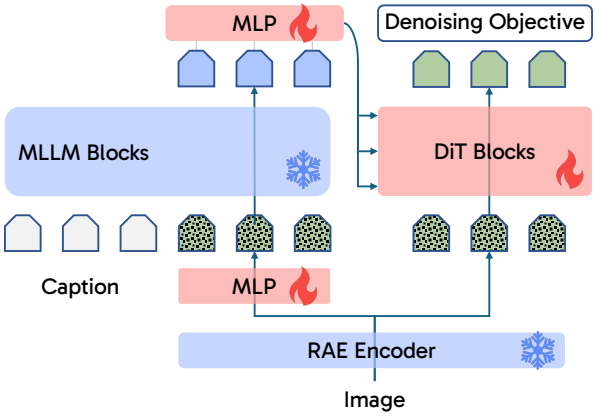


Figure 3: Overview of RepFusion. Blue modules are frozen, while red modules are trainable. We reuse a pretrained MLLM to process the text prompt and noisy RAE latents. The noisy RAE latents are projected into the MLLM input space through an MLP projector, and the resulting outputs condition every DiT block via AdaLN modulation.

Specifically, the LLM input consists of a sequence of text tokens followed by projected noisy visual tokens. Let E_{LLM} denote the LLM, P_ψ an MLP projector, and $\boldsymbol{e}_t$ the timestep embedding; we use the same notation for its projected forms in the visual representation space and the LLM hidden space. The conditioning $\boldsymbol{c}_t$ is defined as

$$\boldsymbol{c}_t = \text{Last}_N\left(E_{\text{LLM}}\left([y, P_\psi(\boldsymbol{z}_t + \boldsymbol{e}_t)]\right)\right) \tag{1}$$

where $[\cdot,\cdot]$ denotes sequence concatenation, and Last_N selects the final N hidden states corresponding to the noisy visual tokens. We inject timestep information before the MLLM by adding the embedding $\boldsymbol{e}_t$ to $\boldsymbol{z}_t$ before applying the projection P_ψ, following Zhou et al. [54]. During training, the MLLM condition and the DiT prediction are computed at the same sampled timestep t. Our default sampling procedure follows this design by running both models at every denoising step, recomputing $\boldsymbol{c}_t$ as $\boldsymbol{z}_t$ evolves. The LLM remains causal.

Following Decoupled Diffusion Transformer (DDT) [45], we condition the DiT using adaptive layernorm [32] without introducing additional cross-attention modules. Appendix C compares

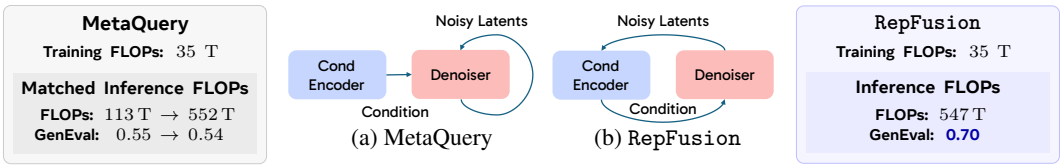


Figure 4: High-level comparison between MetaQuery-style [31] architectures (e.g., BLIP-3o [6] and Scale-RAE [43]) and RepFusion. During training, both methods backpropagate gradients through the conditional encoder and denoiser, resulting in similar training FLOPs. Rerunning the MetaQuery conditional encoder with different timestep embeddings but no noisy visual inputs increases inference compute from 113 to 552 TFLOPs without improving GenEval (0.55 to 0.54). At a similar inference budget, RepFusion reaches 0.70 with conditioning derived from noisy visual representations.

conditioning interfaces. Concretely, we adopt the AdaLN-Single variant from PixArt-α [7]: a shared projection produces token-wise modulation parameters reused across blocks, and each transformer block adds a lightweight learned offset table $\boldsymbol{T}^{(\ell)} \in \mathbb{R}^{6\times D}$ before splitting the resulting parameters into $(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\alpha})$ for the MSA and MLP branches.

Let $\boldsymbol{h}_t^{(\ell)} \in \mathbb{R}^{N\times D}$ denote the intermediate states at DiT block ℓ and timestep t, and let $\boldsymbol{c}_t \in \mathbb{R}^{N\times D_c}$ denote the selected LLM hidden states for the noisy visual tokens. In our setting, the token counts are aligned ($N = 576$). Following DiT [32], we inject the noise level by adding the timestep embedding and applying a SiLU nonlinearity: $\tilde{\boldsymbol{c}}_t = \mathrm{SiLU}(\boldsymbol{c}_t + \boldsymbol{e}_t)$, $\tilde{\boldsymbol{c}}_t \in \mathbb{R}^{N\times D_c}$. A shared linear layer predicts modulation parameters: $\boldsymbol{m}_t = \mathrm{Linear}(\tilde{\boldsymbol{c}}_t) \in \mathbb{R}^{N\times 6D}$. For each block ℓ, we interpret $\boldsymbol{m}_t$ as six token-wise D-dimensional modulation matrices and add a lightweight block-specific table $\boldsymbol{T}^{(\ell)} \in \mathbb{R}^{6\times D}$ (broadcast over tokens) to obtain the final modulation parameters for the block. Splitting along the last channel dimension yields $(\boldsymbol{\beta}^{(\ell)}_{t,\mathrm{msa}}, \boldsymbol{\gamma}^{(\ell)}_{t,\mathrm{msa}}, \boldsymbol{\alpha}^{(\ell)}_{t,\mathrm{msa}}, \boldsymbol{\beta}^{(\ell)}_{t,\mathrm{mlp}}, \boldsymbol{\gamma}^{(\ell)}_{t,\mathrm{mlp}}, \boldsymbol{\alpha}^{(\ell)}_{t,\mathrm{mlp}})$, where each element lies in $\mathbb{R}^{N\times D}$. Here, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ denote the shift and scale terms, respectively, and $\boldsymbol{\alpha}$ denotes the residual gate.

The modulation operator can be defined as $\mathrm{Mod}(\boldsymbol{u}; \boldsymbol{\gamma}, \boldsymbol{\beta}) = \boldsymbol{u} \odot (1 + \boldsymbol{\gamma}) + \boldsymbol{\beta}$. Each block then computes:

$$
\begin{aligned}
\tilde{\boldsymbol{h}} &= \mathrm{Mod}\Big(\mathrm{RMSNorm}(\boldsymbol{h}_t^{(\ell)});\ \boldsymbol{\gamma}^{(\ell)}_{t,\mathrm{msa}},\ \boldsymbol{\beta}^{(\ell)}_{t,\mathrm{msa}}\Big), \\
\boldsymbol{h}_t'^{(\ell)} &= \boldsymbol{h}_t^{(\ell)} + \boldsymbol{\alpha}^{(\ell)}_{t,\mathrm{msa}} \odot \mathrm{MSA}(\tilde{\boldsymbol{h}}), \\
\tilde{\boldsymbol{h}}' &= \mathrm{Mod}\Big(\mathrm{RMSNorm}(\boldsymbol{h}_t'^{(\ell)});\ \boldsymbol{\gamma}^{(\ell)}_{t,\mathrm{mlp}},\ \boldsymbol{\beta}^{(\ell)}_{t,\mathrm{mlp}}\Big), \\
\boldsymbol{h}_t^{(\ell+1)} &= \boldsymbol{h}_t'^{(\ell)} + \boldsymbol{\alpha}^{(\ell)}_{t,\mathrm{mlp}} \odot \mathrm{MLP}(\tilde{\boldsymbol{h}}').
\end{aligned}
$$

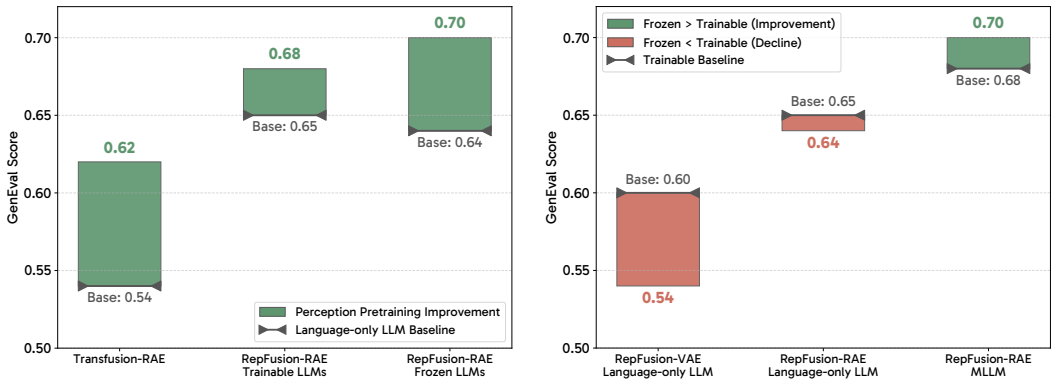
Crucially, $\tilde{\boldsymbol{c}}_t$ is token-aligned with $\boldsymbol{h}_t^{(\ell)}$. As a result, the scale, shift, and residual gates are applied independently to each token rather than being broadcast over the sequence dimension.

MLLM recomputation To reduce sampling cost, we uniformly divide the S denoising steps into U segments, running the MLLM once at the beginning of each segment and reusing its outputs within the segment. The DiT still runs at every denoising step. Unless otherwise specified, all RepFusion results use the default setting $U = S$.

3.3 Noisy representation input

To assess the importance of conditioning the MLLM on noisy representations, as illustrated in Figure 4, we construct a learnable query baseline by replacing the projected noisy RAE latents $P_\psi(\boldsymbol{z}_t + \boldsymbol{e}_t)$ in Equation 1 with N learnable queries $\boldsymbol{Q}_\eta \in \mathbb{R}^{N\times D_c}$, following MetaQuery [31], while keeping the rest of the architecture unchanged. This baseline closely resembles BLIP-3o [6] and Scale-RAE [43]. With a frozen 7B MLLM and a 1.3B DiT, it reaches 0.55 on GenEval, whereas RepFusion reaches 0.70.

Crucially, this gap is not due to additional training compute. RepFusion and the learnable query baseline have similar training FLOPs: for each sampled timestep, both run the same forward pass through the frozen LLM and the denoiser, and both backpropagate through these computations while updating only the conditioning inputs or projector and the denoiser. At inference time, the learnable query baseline computes its condition once, while RepFusion runs both models at every



(a) Effect of multimodal perception pretraining in the LLM backbone. Replacing an LLM with a perception-pretrained MLLM improves both Transfusion-RAE and `RepFusion` under frozen and trainable-LLM settings.

(b) Effect of fine-tuning the LLM backbone. Fine-tuning helps when starting from a language-only LLM, but can hurt when starting from a perception-pretrained MLLM backbone in `RepFusion`-RAE.

Figure 5: Ablations on multimodal perception pretraining and LLM fine-tuning. Perception-pretrained backbones consistently improve RAE diffusion, while fine-tuning benefits text-only backbones but may degrade performance when the backbone is already multimodally pretrained. This indicates that perception pretraining is a strong prior that can outperform joint optimization for generation.

denoising step to condition on the current z_t. To test whether the gain can be explained by additional inference compute, we make the learnable queries timestep-dependent and rerun their encoder at every denoising step, closely matching `RepFusion` in inference FLOPs. This variant reaches only 0.54 on GenEval, below the original learnable query baseline. This comparison supports the value of an input-dependent conditioning signal over evolving noisy representations.

3.4 Multimodal perception pretraining

The gains above depend on the conditional encoder being able to interpret structured visual representations along a denoising trajectory. We therefore examine what makes an LLM better at interpreting noisy RAE latents.

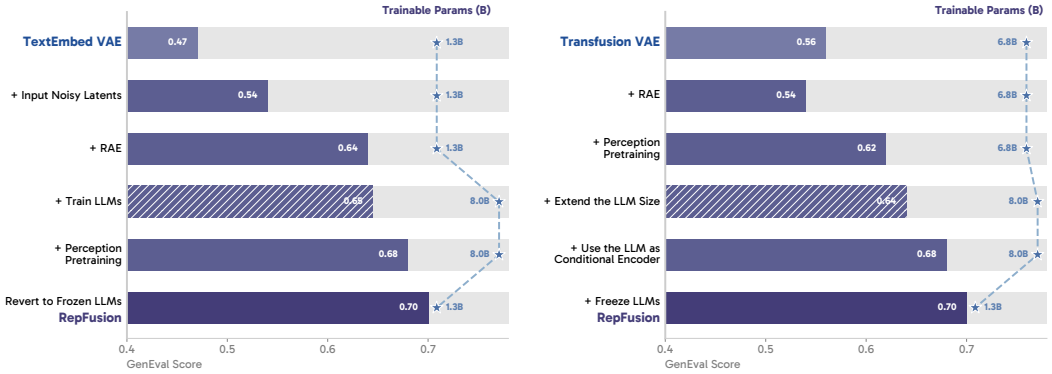
In particular, our conditional encoder is an MLLM whose backbone has been pretrained to perceive visual representations. We therefore study the role of multimodal perception pretraining. To isolate the effect of this capability, we compare a language-only LLM backbone with a perception-pretrained MLLM backbone, while keeping the denoiser and token budget the same. As shown in Figure 5a, replacing the language-only LLM with a perception-pretrained MLLM improves both Transfusion-RAE and `RepFusion` under settings with frozen and trainable LLMs. This indicates that perception pretraining provides a transferable prior for diffusion in RAE space: an MLLM that can interpret clean visual representations also better supports encoding their noisy counterparts.

We further compare preserving a perception-pretrained LLM with jointly optimizing the LLM for generation. Following Transfusion [54], when fine-tuning the LLM, we add an auxiliary language modeling (LM) loss on the caption tokens and allow the injected noisy visual tokens to use bidirectional attention, while keeping the caption stream causal. Figure 5b shows a consistent pattern: fine-tuning helps when starting from a language-only LLM backbone (in both VAE and RAE setups), but it degrades performance when starting from a perception-pretrained MLLM backbone in `RepFusion`-RAE. This suggests that multimodal perception pretraining is a strong prior that is best preserved rather than further re-optimized for generation.

3.5 Breaking down the improvement

We use these observations to break down the gains of `RepFusion` over TextEmbed and Transfusion.

As shown in Figure 6a, starting from a standard text embedding baseline with a GenEval score of 0.47, feeding noisy VAE latents into the LLM improves the score to 0.54. Replacing VAE latents with RAE latents, which are easier to denoise and more compatible with LLMs, further improves the



(a) Path from TextEmbed to `RepFusion`.

(b) Path from Transfusion to `RepFusion`.

Figure 6: Step-by-step ablations from (a) TextEmbed and (b) Transfusion toward `RepFusion`. Bars show GenEval scores, and hatched bars denote modifications that are evaluated but not adopted.

score to 0.64. Jointly training the LLM with an LM loss and bidirectional attention gives a minor gain to 0.65. Adding multimodal perception pretraining improves denoising in RAE space, but the best performance is achieved when the LLM backbone remains frozen, preserving the pretrained prior.

We also trace the path from the Transfusion baseline. Transfusion can be viewed as another way of exposing noisy visual latents to language models, but this unified baseline can be improved by using the LLM explicitly as a conditional encoder rather than as the denoiser itself. As shown in Figure 6b, starting from the Transfusion baseline at 0.56, replacing VAE latents with RAE latents and adopting a perception-pretrained LLM improves performance to 0.62. We then replicate the last 6 layers of the LLM to construct a stronger 8.0B Transfusion-RAE baseline. Reallocating the same 1.3B trainable parameters to a separate DiT, while using the LLM as the conditional encoder, improves performance from 0.64 to 0.68; freezing the LLM further increases it to 0.70.

4 Experiments

4.1 Experimental setup

Model Unless otherwise specified, we follow the MLLM setup of Liu et al. [26]: a causal LLM backbone is paired with a CLIP-L/14 vision tower [34] through an MLP projector, which provides a clean interface for our RAE setup. We follow this simple architecture because many recent MLLMs introduce fine-tuned vision towers, any-resolution support, token compression [41], and deep stacks [2], which are tailored for multimodal understanding and are non-trivial to adopt for denoising purposes. We set the input resolution to 336, producing $N{=}576$ visual tokens. For VAE-based experiments, we use DC-AE [9] with a spatial downsampling factor of 32. We set the input resolution to 512, which yields $N{=}256$ latent tokens. This setup keeps output resolutions comparable across different latent spaces; we include a token-matched DC-AE comparison in Appendix D. For both RAE and VAE settings, we set the DiT patch size to 1.

Data We pretrain all models on the BLIP-3o 31M dataset [6], which is recaptioned with MLLMs and contains 27M long- and 4M short-caption pairs. For supervised fine-tuning (SFT), we combine BLIP-3o 60k [6], ShareGPT4o-Image [8], and Echo-4o [51] into a 200k synthetic dataset. SFT images are sourced from GPT-4o Image [30].

Training For all pretraining experiments, we train the models on 128 H200 GPUs with a global batch size of 2,048. Models are trained for 10 epochs (160k steps) with a learning rate of 3×10^{-4}. They are optimized using the AdamW [28] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a weight decay of 0.1. The learning rate follows a cosine decay schedule with a 10k-step warmup period. For SFT experiments, we use a learning rate of 1×10^{-4} and train the model for 64 epochs.

Representation Decoders To decode visual representations into pixels, we use two strategies: an RAE decoder [53] and a diffusion decoder [40]. Unless otherwise specified, all parameter counts

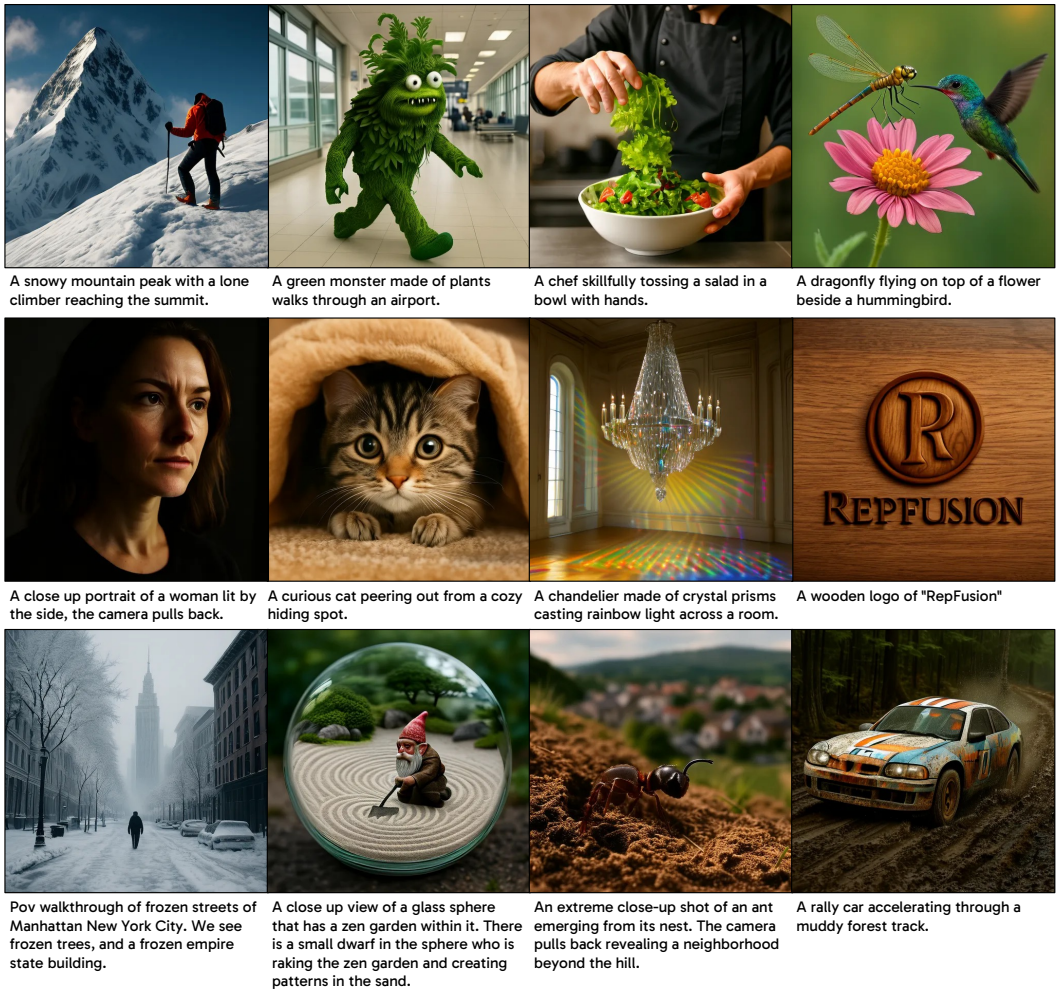


Figure 7: Qualitative T2I samples generated by `RepFusion`. Some prompts are from MovieGen [33].

reported exclude decoder parameters, and the decoder is the RAE decoder by default. For the RAE decoder, we follow standard RAE practice by using a ViT-XL [17] decoder and a DINO [5] GAN discriminator. The decoder patch size is set to 24 to produce images at a resolution of 576. We train the RAE decoder on ImageNet-22k [11] for 16 epochs. For the diffusion decoder, we follow Emu [40], starting from the SANA 1.6B checkpoint and replacing the text conditioning with CLIP features. The output resolution is 512. We train the diffusion decoder on ImageNet-22k for 10 epochs.

4.2 Prompt alignment

With only around 30M image-caption pairs, `RepFusion` achieves strong T2I prompt alignment, with qualitative samples shown in Figure 7. We evaluate our largest configuration, which uses a 7B MLLM and a 3.2B DiT, on four representative benchmarks: GenEval [13], GenEval++ [51], GenEval2 [21], and DPG-Bench [20]. As shown in Table 1, `RepFusion` achieves competitive performance across all of them, and `RepFusion`-SFT further improves the performance to state-of-the-art levels. Notably, benchmarks such as GenEval and DPG-Bench are increasingly subject to benchmark-specific optimization in the current synthetic data era [49]: many pipelines perform SFT on synthetic images sourced from GPT-4o [30] and Nano Banana [15], or directly apply RL with GenEval as a verifiable reward. To address this benchmark drift issue, GenEval2 was recently proposed with a more robust evaluation protocol, Soft-TIFA. Consistent with this motivation, we find that `RepFusion`-SFT yields only limited improvements on GenEval2, while the pretrained `RepFusion` remains strong. `RepFusion` also compares favorably with BAGEL [10] with Self-CoT, which is pretrained on over 1 billion web-scale examples.

Table 1: T2I generation results on GenEval [13], GenEval++ [51], GenEval2 [21], and DPG-Bench [20]. † denotes rewritten prompts. For GenEval2, we report prompt-level Soft-TIFA$_{GM}$.

Methods	GenEval ↑	GenEval++ ↑	GenEval2 ↑	DPG-Bench ↑
Transfusion [54]	0.63	-	-	-
MetaQuery-XL [31]	0.80†	-	-	82.05
BLIP-3o 8B [6]	0.84	0.307	-	81.60
OmniGen2 [47]	0.80	0.325	-	83.57
BAGEL [10]	0.82	0.371	23.1†	84.03
Scale-RAE [43]	0.83	-	-	79.70
RepFusion w/ RAE Decoder	0.73	0.432	30.2	82.75
RepFusion w/ Diffusion Decoder	0.78	0.443	29.9	84.41
RepFusion-SFT w/ RAE Decoder	0.85	0.707	35.1	84.17
RepFusion-SFT w/ Diffusion Decoder	0.87	0.669	34.9	85.11

Table 2: Reasoning-based generation on WISE [29].

Model	Cultural	Time	Space	Biology	Physics	Chemistry	Overall
MetaQuery-XL [31]	0.56	0.55	0.62	0.49	0.63	0.41	0.55
BLIP-3o 8B [6]	-	-	-	-	-	-	0.62
BAGEL [10]	0.44	0.55	0.68	0.44	0.60	0.39	0.52
RepFusion-SFT w/ RAE Decoder	0.55	0.53	0.70	0.51	0.57	0.41	0.55
RepFusion-SFT w/ Diffusion Decoder	0.65	0.63	0.79	0.63	0.67	0.44	0.64

4.3 Reasoning-based generation

We find that, similar to learnable query methods such as MetaQuery [31] and BLIP-3o [6], RepFusion can also effectively leverage the capabilities of a frozen LLM. This enables the model to better understand and follow complex prompts, including those requiring world knowledge and reasoning. To quantitatively evaluate RepFusion's world-knowledge reasoning capability, we employ the WISE [29] benchmark. As shown in Table 2, RepFusion matches state-of-the-art performance.

4.4 Scaling behavior

RepFusion has two main components: the frozen MLLM that encodes noisy representations, and the DiT denoiser that predicts the velocity. We study how performance changes when scaling either component in the billion-parameter regime, using per-step MLLM recomputation throughout. Figure 8 shows that increasing either the MLLM or DiT size can improve performance, with the clearest scaling trends appearing on GenEval and GenEval++.

We further study how to allocate inference compute across these two components under iso-FLOPs settings in Table 3. At around 280T FLOPs, allocating more compute to the DiT outperforms the configuration that allocates more compute to the MLLM across all metrics. The same trend holds at around 540T FLOPs: configurations with larger DiTs achieve stronger GenEval++ and GenEval2 scores. These within-family comparisons answer a different question from Figure 2: among RepFusion variants, scaling the DiT is generally more effective under a fixed inference budget, but compared with TextEmbed, RepFusion remains stronger even when the baseline spends nearly all sampling compute on the DiT. Thus, pretrained MLLM priors complement denoiser capacity, improving generation without eliminating the benefits of scaling the DiT.

4.5 MLLM recomputation

The RepFusion results above use per-step MLLM recomputation. We now examine whether these pretrained priors can be used with less inference compute. Using RepFusion w/ RAE Decoder from Table 1, we reduce the number of MLLM updates U while keeping the trained model and 30 denoising steps unchanged. Table 4 shows that three updates retain performance comparable to per-step recomputation (0.733 vs. 0.730 on GenEval), with a 2.43× sampling speedup. Qualitative comparisons are provided in Appendix F.

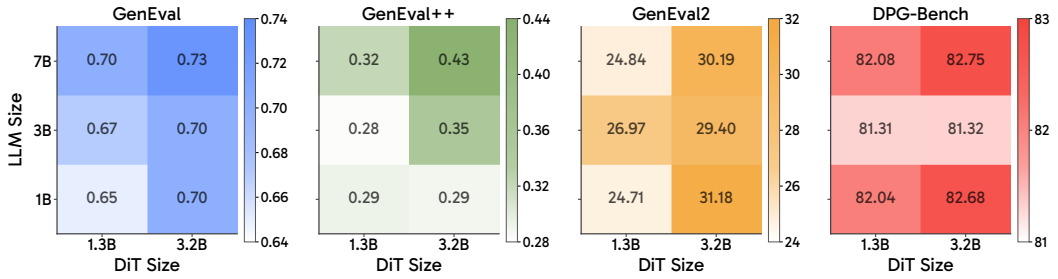


Figure 8: MLLM and DiT co-scaling results. Increasing either component can improve performance, with clearer trends on GenEval and GenEval++.

Table 3: Iso-FLOPs comparison. Given a fixed inference budget, we compare different allocations between MLLMs and DiTs within `RepFusion`, and include TextEmbed as a reference baseline. Scaling the DiT is generally more favorable than scaling the MLLM within `RepFusion`, but `RepFusion` remains substantially stronger than TextEmbed, whose static text embedding design leaves nearly all sampling compute in the DiT.

LLM Size	DiT Size	FLOPs Split (LLM | DiT)	GenEval ↑	GenEval++ ↑	GenEval2 ↑	DPG-Bench ↑
∼280T inference FLOPs						
1B	3.2B	26% | 74%	0.70	0.289	31.18	82.68
3B	1.3B	71% | 28%	0.67	0.282	26.97	81.31
∼540T inference FLOPs						
7B (TextEmbed baseline)	8.0B	3% | 97%	0.66	0.321	26.60	81.34
1B	7.3B	13% | 87%	0.70	0.443	30.84	82.58
3B	5.5B	37% | 63%	0.69	0.382	30.67	82.20
7B	1.3B	85% | 15%	0.70	0.321	24.84	82.08

Table 4: Reducing MLLM recomputation during inference. Latency is measured on a single H200 GPU. The $U = 30$ row corresponds to the `RepFusion` w/ RAE Decoder result in Table 1. Three MLLM updates achieve performance comparable to per-step MLLM recomputation, with a 2.43× sampling speedup.

MLLM Updates U	FLOPs (T)	Latency (s)	Speedup	GenEval ↑
30	665	1.43	1.00×	0.730
15	440	0.97	1.48×	0.730
6	306	0.69	2.07×	0.725
3	261	0.59	2.43×	0.733
1	231	0.52	2.73×	0.694

The $U = 1$ setting also provides a reference for inference cost: the MLLM runs once and the DiT runs for all 30 denoising steps, as in TextEmbed. For the same model, three updates require only 13% more inference FLOPs than this single-update setting (261T vs. 231T). Together with our controlled comparisons against TextEmbed in Figures 1 and 2, these results suggest that noisy visual conditioning can provide substantially stronger generation while approaching the inference cost of computing the condition once, as in TextEmbed.

5 Conclusion

We study a simple but underused degree of freedom in modern T2I systems: the conditional encoder. By allowing a frozen MLLM to read noisy visual representations, the encoder becomes an active component of the denoising loop rather than a static text encoder. The resulting gains suggest a practical recipe for future models: expose pretrained MLLM priors to evolving noisy representations and preserve those priors. We hope this perspective helps shift how the community thinks about the role of MLLMs in T2I generation.

Acknowledgments and Disclosure of Funding

This work was conducted during Xichen Pan's internship at Meta, which provided computing resources and internship compensation. The authors report no additional funding or competing interests related to this work beyond the support and institutional affiliations stated here.

References

[1] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *ICLR*, 2023.

[2] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[4] Huanqia Cai, Sihan Cao, Ruoyi Du, Peng Gao, Steven Hoi, Zhaohui Hou, Shijie Huang, Dengyang Jiang, Xin Jin, Liangchen Li, et al. Z-image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025.

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[6] Jiuhai Chen, Zhiyang Xu, Xichen Pan, Yushi Hu, Can Qin, Tom Goldstein, Lifu Huang, Tianyi Zhou, Saining Xie, Silvio Savarese, et al. Blip3-o: A family of fully open unified multimodal models-architecture, training and dataset. *arXiv preprint arXiv:2505.09568*, 2025.

[7] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. Pixart-alpha: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *ICLR*, 2024.

[8] Junying Chen, Zhenyang Cai, Pengcheng Chen, Shunian Chen, Ke Ji, Xidong Wang, Yunjin Yang, and Benyou Wang. Sharegpt-4o-image: Aligning multimodal models with gpt-4o-level image generation. *arXiv preprint arXiv:2506.18095*, 2025.

[9] Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In *ICLR*, 2025.

[10] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[12] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024.

[13] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In *NeurIPS*, 2023.

[14] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[15] Google. Introducing gemini 2.5 flash image, our state-of-the-art image model, 2025. URL `https://developers.googleblog.com/introducing-gemini-2-5-flash-image/`.

[16] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[17] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Neural computation*, 1997.

[20] Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024.

[21] Amita Kamath, Kai-Wei Chang, Ranjay Krishna, Luke Zettlemoyer, Yushi Hu, and Marjan Ghazvininejad. Geneval 2: Addressing benchmark drift in text-to-image evaluation. *arXiv preprint arXiv:2512.16853*, 2025.

[22] Diederik P Kingma. Auto-encoding variational bayes. In *ICLR*, 2014.

[23] Black Forest Labs. Flux.1. `https://bfl.ai/blog/24-08-01-bfl`, 2024.

[24] Black Forest Labs. FLUX.2: Frontier Visual Intelligence. `https://bfl.ai/blog/flux-2`, 2025.

[25] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2023.

[26] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[27] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[29] Yuwei Niu, Munan Ning, Mengren Zheng, Bin Lin, Peng Jin, Jiaqi Liao, Kunpeng Ning, Bin Zhu, and Li Yuan. Wise: A world knowledge-informed semantic evaluation for text-to-image generation. In *ICML*, 2026.

[30] OpenAI. Introducing 4o image generation, 2025. URL `https://openai.com/index/introducing-4o-image-generation/`.

[31] Xichen Pan, Satya Narayan Shukla, Aashu Singh, Zhuokai Zhao, Shlok Kumar Mishra, Jialiang Wang, Zhiyang Xu, Jiuhai Chen, Kunpeng Li, Felix Juefei-Xu, et al. Transfer between modalities with metaqueries. *arXiv preprint arXiv:2504.06256*, 2025.

[32] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

[33] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

[34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[35] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.

[36] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.

[37] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[38] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022.

[39] Weijia Shi, Xiaochuang Han, Chunting Zhou, Weixin Liang, Xi Victoria Lin, Luke Zettlemoyer, and Lili Yu. Lmfusion: Adapting pretrained language models for multimodal generation. In *NeurIPS*, 2025.

[40] Quan Sun, Qiying Yu, Yufeng Cui, Fan Zhang, Xiaosong Zhang, Yueze Wang, Hongcheng Gao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. Generative pretraining in multimodality. In *ICLR*, 2024.

[41] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[42] Mistral AI Team. Mistral small 3, 2025. URL `https://mistral.ai/news/mistral-small-3`.

[43] Shengbang Tong, Boyang Zheng, Ziteng Wang, Bingda Tang, Nanye Ma, Ellis Brown, Jihan Yang, Rob Fergus, Yann LeCun, and Saining Xie. Scaling text-to-image diffusion transformers with representation autoencoders. In *CVPR*, 2026.

[44] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[45] Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. In *CVPR*, 2026.

[46] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

[47] Chenyuan Wu, Pengfei Zheng, Ruiran Yan, Shitao Xiao, Xin Luo, Yueze Wang, Wanli Li, Xiyan Jiang, Yexin Liu, Junjie Zhou, et al. Omnigen2: Exploration to advanced multimodal generation. In *CVPR*, 2026.

[48] Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. In *ICLR*, 2025.

[49] Ji Xie, Trevor Darrell, Luke Zettlemoyer, and XuDong Wang. Reconstruction alignment improves unified multimodal models. In *ICLR*, 2026.

[50] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, 2018.

[51] Junyan Ye, Dongzhi Jiang, Zihao Wang, Leqi Zhu, Zhenghao Hu, Zilong Huang, Jun He, Zhiyuan Yan, Jinghua Yu, Hongsheng Li, et al. Echo-4o: Harnessing the power of gpt-4o synthetic images for improved image generation. *arXiv preprint arXiv:2508.09987*, 2025.

[52] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[53] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *ICLR*, 2026.

[54] Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. In *ICLR*, 2025.

[55] Le Zhuo, Ruoyi Du, Han Xiao, Yangguang Li, Dongyang Liu, Rongjie Huang, Wenze Liu, Lirui Zhao, Fu-Yun Wang, Zhanyu Ma, et al. Lumina-next: Making lumina-t2x stronger and faster with next-dit. In *NeurIPS*, 2024.

A Limitations

With per-step MLLM recomputation, `RepFusion` is more expensive than reusing a static text embedding, but comparable to baselines that allocate similar parameters to modules that process noisy visual latents or denoise them. Reusing MLLM outputs across denoising steps reduces this overhead: three updates over 30 steps retain comparable performance with a 2.43$\times$ sampling speedup (Section 4.5). Because the MLLM input contains sample-specific noisy latents, its visual-token outputs cannot be shared across same-prompt samples. Our experiments also focus on a controlled MLLM interface and a fixed 31M-pair training set; extending the study to newer MLLM families and larger data scales remains future work. Decoder-dependent evaluation effects are discussed in Appendix E.

B Details on training TextEmbed and Transfusion baselines

We train the TextEmbed and Transfusion baselines with the same training setup as `RepFusion`. The controlled comparisons use the same text encoder family and newly initialized denoising components at the corresponding model scale. For TextEmbed, we follow the recent T2I practice used in Sana [48]: the LLM is used as a static text encoder, and its last-layer text-token embeddings condition a newly initialized DiT denoiser. Following Sana, we also apply RMSNorm after the decoder-only text encoder to normalize the variance of the text embeddings to 1.0. For Transfusion [54], in addition to the diffusion training described above, we perform interleaved image-captioning training for the same number of training steps, using the same image-caption data.

C Conditioning interface

We ablate how the MLLM hidden states are injected into the DiT. Cross attention provides a general conditioning mechanism, but it adds extra attention projections and treats the conditioning stream as a separate context. In our RAE setting, the N MLLM outputs are naturally aligned with the N DiT tokens, so a token-wise adaptive normalization interface can use this correspondence directly. As shown in Table 5, AdaLN-Single [7] achieves a slightly higher GenEval score with fewer parameters, and we therefore use it as the default conditioning interface.

Table 5: Ablation on the interface used to inject MLLM hidden states into the DiT. Parameter counts include the DiT and the conditioning interface.

Method	**# Params**	**GenEval** $\uparrow$
Cross Attention	1.6B	0.69
AdaLN-Single [7]	1.3B	0.70

D Additional latent space comparison

In the main experiments, we keep the output resolutions comparable across latent spaces, which gives N=256 tokens for DC-AE and N=576 tokens for RAE. To isolate the effect of token count, we also evaluate a DC-AE setting with N=576 tokens by increasing its output resolution. As shown in Table 6, matching the DC-AE token count does not improve the TextEmbed baseline.

Table 6: Token-matched latent-space comparison for TextEmbed. Increasing the DC-AE sequence length to match the RAE setting does not close the gap to RAE latents.

Method	**Latent Space**	**Resolution**	**Sequence Length**	**GenEval** $\uparrow$
TextEmbed	DC-AE	768	576	0.45
TextEmbed	DC-AE	512	256	0.47
TextEmbed	RAE	576	576	0.57

E Impact of decoders

We report `RepFusion` results with both the RAE decoder and the diffusion decoder in Table 1. We observe a performance gap between these two decoders. However, when we use `RepFusion` to

generate a CLIP feature and then apply these two decoders to the same feature, the resulting images appear very similar. The overall layout and colors are largely determined by the CLIP feature, while only fine-grained textures differ slightly (Figure 9). This suggests that the choice of decoder does not affect the prompt-following ability of `RepFusion`. Instead, part of the performance gap on GenEval and DPG-Bench appears to arise because images decoded by the RAE decoder are blurrier in texture and therefore harder for detectors or vision-language models (VLMs) to evaluate reliably.



Figure 9: Same CLIP representation decoded by the diffusion decoder and the RAE decoder. Since both decoders are optimized for reconstruction, the mapping from CLIP features to pixels is largely deterministic: object layout and colors are determined upstream when the model denoises the CLIP representation. The choice of decoder mainly affects fine-grained appearance (e.g., textures), and thus has little impact on object-level prompt-following.

F Qualitative comparison of MLLM recomputation

Figure 10 complements Section 4.5 with samples generated using $U = 30$, 6, 3, and 1 MLLM updates over 30 denoising steps. Each group uses the same prompt and initial-noise seed. The DiT runs at every step, reusing the latest MLLM outputs between updates. The samples remain visually similar with $U = 30$, 6, and 3, but change noticeably at $U = 1$.

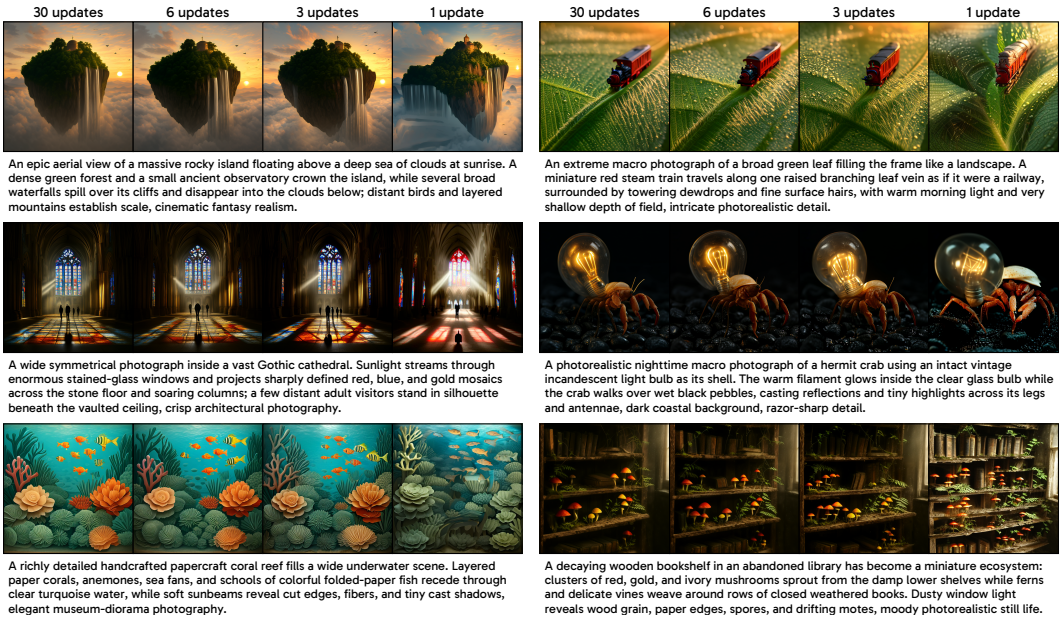


Figure 10: Qualitative comparison of MLLM recomputation. Within each group, the four columns share the same initial-noise seed and use 30, 6, 3, and 1 MLLM updates over 30 denoising steps.

G License

- LLaVA-1.5-7B / Vicuna-7B-v1.5 [26]: Llama 2 Community License Agreement.
- Llama 3.2 [16]: Llama 3.2 Community License Agreement.
- CLIP-L/14 [34]: MIT License.
- DC-AE [9]: Apache License 2.0.
- Sana 1.6B, 512px [48]: Apache License 2.0.
- BLIP-3o 31M and BLIP-3o 60k [6]: Apache License 2.0.
- ShareGPT4o-Image [8]: Apache License 2.0.
- Echo-4o-Image [51]: MIT License.
- ImageNet-22k [11]: ImageNet non-commercial research and educational use terms.
- GenEval [13]: MIT License.
- GenEval++ [51]: Apache License 2.0.
- GenEval2 [21]: CC BY-NC 4.0.
- DPG-Bench [20]: Apache License 2.0.
- WISE [29]: license not specified.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main claims, including noisy-representation conditioning and test-time scaling.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix A discusses the main limitations, including efficiency and decoder-dependent evaluation effects.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not make theoretical claims or present theorem-style results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The architecture and objective are described in Section 3, and the training data, model setup, optimization details, and decoder setup are described in Section 4.1. The main evaluation protocols and results are reported in Tables 1–3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We do not provide open access to the training code or processed data in this submission. The experiments use public datasets and benchmarks cited in Section 4.1.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Section 4.1 specifies the model setup, datasets, optimizer, learning rates, batch size, training length, and decoder settings. The benchmark settings and reported metrics are summarized in Sections 4.2–4.4.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We report point estimates on standard benchmarks but do not include error bars or confidence intervals. Repeating all large-scale training runs would be computationally expensive.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4.1 reports the GPU type, number of GPUs, batch size, training length, optimization settings, and inference FLOPs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: To the best of our knowledge, the research conforms to the NeurIPS Code of Ethics. The work uses public datasets and benchmarks.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper does not include a dedicated broader-impact discussion, and this submission does not release new models or datasets. Potential misuse risks are those broadly associated with T2I generation rather than risks specific to the proposed method.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release new datasets or model checkpoints in this submission. Any future release should follow appropriate access, content-safety, and misuse-mitigation policies for image generation models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the public datasets, benchmarks, and model components used in the paper and list their licenses in Appendix G.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce or release a new dataset or other standalone asset. All training data used in the experiments comes from public existing sources described in Section 4.1.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: The paper does not involve research with human subjects, so IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.